Inverse Reinforcement Learning: a New Framework to Mitigate an Intelligent Backoff Attack

Juan Parras, Alejandro Almodóvar, Patricia A. Apellániz, and Santiago Zazo

***Abstract*—The recent advances in Deep Learning have a significant impact on the security of wireless networks, such as intelligent attackers which are able to successfully exploit a possibly unknown defense mechanism simply by interacting with it. Their capacity of adapting to standard defense mechanisms, such as statistical tests, make them a significant threat. In this paper, we develop two intelligent defense mechanisms using inverse reinforcement learning tools, that can be used to enhance the capabilities of current defense mechanisms. We test our proposal on a backoff attack setup against an intelligent attacker, obtaining very significant gains in the defense performance.**

***Index Terms*—Inverse Reinforcement Learning, Markov Decision Process, Wireless Network, Security, Artificial Intelligence**

I. Introduction

TODAY, a lot of effort is devoted to the research on wireless networks [1], [2], [3], where one of the main topics addressed is security. The increase in the number of interconnected devices and the increasing number of applications available to users has also brought an important increase in the number of vulnerabilities. Whereas recent research tries to include security in wireless network protocols, a recent work states that many of these solutions are still at a proof-of-concept level [4], and other authors claim that most defense solutions are restricted to a few types of attacks [5].

In parallel, a field that has experienced a huge development in the past few years is Deep Learning [6], whose advances have also been applied to network security [7], [8]. A particular Artificial Intelligent field that has taken advantage of the research in Deep Learning is Reinforcement Learning (RL). RL is used to solve problems in which decisions are made sequentially, and it tries to obtain an optimal policy, i.e., an optimal control law, which defines the best action to take at each moment. Currently, RL tools are used in several problems in wireless networks, such as routing, data latency, path determination, duty cycle management, QoS provisioning, or resource management [9]. Besides, RL has been applied to enhance security in wireless networks [10], [11], and more recently, Deep RL (DRL) methods, that combine RL tools and Deep Learning advances, have been proposed for attack detection [12], [13], for mobile offloading [14], [15] or for jamming avoidance [16], [17].

The potential impact of DRL methods in security applications is important. We note that the literature on attack detection is extensive, with many papers ranging from general works, as [4], [18] or [19], to works specialized in concrete attacks, such as byzantine attacks [20], [21], DOS attacks [22], jamming situations [23] or backoff attacks [24]. But all of these defense mechanisms have a common point: since they have been designed against a concrete attack, they all use a priori knowledge about that expected attack technique. Hence, if the attacker is able to learn the weaknesses of a concrete defense mechanism, it would be able to exploit it by changing its attack strategy. Note that this idea is at the core of the security field: novel attacks that exploit vulnerabilities of current defense mechanisms are proposed, and in response, new defense mechanisms that address these vulnerabilities are developed. This process involves carefully handcrafted strategies both for attack and defense, involving human intervention. However, DRL methods can make the attack strategy design automatic, as they are able to obtain the optimal attack strategies simply by interacting with the defense mechanism. Through this paper, we will use the term “intelligent” to describe attackers able to modify their attack strategy depending on the defense mechanism strategy, in order to exploit it as optimally as possible: these intelligent attackers can be obtained using DRL tools [25], [26]. As noted in [5], current defense mechanisms would not be able to cope with such intelligent attackers.

Thus, given the potential of RL tools to create intelligent attackers, it is surprising that they are not well assessed yet on current literature. This is shared by the authors of [27], and they hypothesize that this may be due to the fact that DRL techniques face the problems of sample efficiency and reward modelling, which, they propose, could be overcome by training the DRL agent using a simulator. A notable exception is the smart-grid field, where several recent works propose using RL to design attack methods [28], [29], [30] and defense mechanisms [31], [32], [33]. It is interesting noting that, even in these works, it is frequently assumed that the attacker is not intelligent, e.g., [31] detects attacks on a smart-grid setup using RL tools, but these attacks have fixed equations that characterize them. It has also been reported that RL techniques are successful in attacking a crowdsensing system [34], and more important to our work, RL attackers are also successful against defense mechanisms used in WSNs [25], [26].

Thus, intelligent attackers based on DRL are a significant threat to current defense mechanisms that are not well assessed yet in literature. In this paper, we propose using DRL tools also to design a defense mechanism that is successful against such attackers. The key idea is to design a defense mechanism that makes as few assumptions as possible about the attack technique, so that it is able to detect a wide range of different attacks. More concretely, we propose using Inverse RL (IRL) tools in order to design defense mechanisms that are able to successfully deal with intelligent attacks, such as those based on DRL tools. Our main contributions are:

- We use IRL tools in order to design two defense mechanisms able to cope with intelligent attacks based on DRL in wireless networks. These tools allow detecting an attack based on the difference of behavior with regards to the case without attack, operating in conditions of partial observability, which are frequent in real-life problems.
- By not assuming a concrete attack type, our methods are able to potentially detect a broad set of attacks.
- By taking advantage of recent developments in the Deep Learning field, our defense mechanisms are able to deal with high dimensionality problems in an automatic fashion that requires little tuning.
- We test our defense mechanisms in a backoff attack setup, which may potentially affect any wireless network with CSMA/CA multiple access method. The effects of this attack are that some devices achieve a larger share of the network resources [35]: as we will see, our defense mechanisms are able to successfully

J. Parras, A. Almodóvar, P. A. Apellániz and S. Zazo are with Information Processing and Telecommunications Center, Universidad Politécnica de Madrid, ETSI Telecomunicación, Av. Complutense 30, 28040 Madrid, Spain.

Manuscript received April 19, 2005; revised August 26, 2015.

counter this attack.

All these advantages make our defense mechanisms an important step against intelligent attackers. The two closest works to ours in current literature are [27] and [5]. In [27], the authors use RL for attack mitigation using an SDN. They consider their work a proof-of-concept to show that RL methods can be successfully used in defense mechanisms, but their ideas limit to perfect observability environments with attack techniques known a priori. In [5], the authors also use RL for detecting attackers which have more variability, as their attackers may choose different threshold attacks. In that setting, they use RL methods to design a defense mechanism that has a good performance. However, these methods are restricted to the perfect observability situation, and also, the attack techniques are somehow limited, as none of their attacks has the intelligence of a DRL attacker. On the contrary, our work presents a defense mechanism that is not only able to operate in partial observability conditions, but also it is able to counter an intelligent attacker, based on DRL.

The rest of the paper goes as follows: in Section II, we introduce the RL theoretical framework in which our work is based on. Then, Section III introduces key IRL concepts and methods. After, Section IV introduces the backoff attack problem that we use as test-bench and Section V explains the DRL intelligent attacker architecture. Then, Section VI, introduces our two defense mechanisms based on IRL tools, whose performance is tested in Section VII. Finally, some conclusions are drawn in Section VIII.

II. Theoretical framework

Let us introduce the discrete time Markov Decision Process (MDP) framework, which is a flexible, well-studied and widely used model to describe dynamical systems [36], [37], [38]. We rely on this framework in order to model the attack setup.

A. Markov Decision Process

An MDP is defined as follows [36], [38]:

Definition 1. *An MDP is a 5-tuple $\langle S, A, P, R, \gamma\rangle$ where:*

- *S is the state set, containing all the possible states $s \in S$ of the dynamical system.*
- *A is the action set, containing all the possible action vectors $a \in A$ that the agent can use to interact with the dynamical system.*
- *$P : S \times S \times A \to [0, 1]$ is the transition probability function, where $P(s'|s, a)$ denotes the probability of transitioning to state s' given that the agent is in state s and takes action a.*
- *$R : S \times A \to \mathbb{R}$ is the reward function, where $r(s, a)$ denotes the reward that the agent receives when it is in state s and takes action a. We assume that R is bounded.*
- *$\gamma \in (0, 1)$ is a discount factor.*

In general, MDPs can be of finite or infinite horizon, depending on whether the final time N is finite or infinite. As noted in [36], the infinite horizon problem never holds in practice, however, it is a reasonable approximation for problems with many time steps. In this work we use methods for solving MDPs in infinite horizon because these scale better for large state and action spaces compared to methods for finite horizon cases [36].

A key idea behind an MDP is the Markovian property: the probability of transitioning to state s' by playing action a depends only on the current state s and is independent of previous states. Thus, the solution for an infinite horizon MDP is a stationary policy $\pi : S \to A$, which is a probability distribution over A denoting the probability that the agent plays action $a \in A$ where it is in state s.

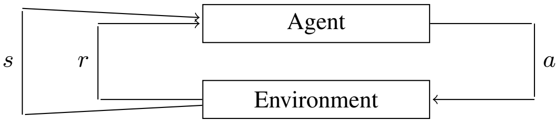


Fig. 1. MDP basic interaction scheme.

An MDP has the cyclic behavior shown in Figure 1. There is a single agent that interacts with the dynamical system. At each time step n, the system is in a certain state s known by the agent, who plays an action a following a certain policy π. The system transitions to state s' and the agent receives a reward $r(s, a)$.

B. Solving an infinite horizon MDP

In an infinite horizon problem, it is common to define the value function $V_\pi(s)$ as a mapping $V_\pi : S \to \mathbb{R}$ that represents the expected total reward when the agent starts in a state s and follows the policy π as follows:

$$V_\pi(s) = \mathbb{E}_{\pi,P}\left[\sum_{n=0}^{\infty}\gamma^n r^n | s, a \sim \pi\right], \tag{1}$$

where we use the shorthand $r^n = r(s, a)$ for time step n. Note that the discount factor $\gamma \in (0, 1)$ is used to weight rewards.

When P is known, it is possible to apply Dynamic Programming tools in order to obtain the optimal policy for an MDP, i.e., the policy that maximizes $V(s) \quad \forall s \in \mathcal{S}$ [37], [39]. However, it is frequent that P is unknown; in this case, model-free RL methods, which approximate an optimal policy by interacting with the dynamical system, can be used instead.

C. Reinforcement Learning

RL methods are biologically inspired and their basic intuition is that it is possible to learn how to act optimally in a dynamical system like the one shown in Figure 1 by using trial and error. Thus, the agent stores data about its interactions in tuples (s, a, r, s'). This data is then used by the agent to optimize a policy π that maximizes the value (1). A complete introduction to the field is given in [39].

Recent advances in the Deep Learning field [6] have impacted significantly RL algorithms. Deep Neural Networks (DNNs), which are universal function approximators [40], [41], have been used to create DRL methods that scale to large state and/or action spaces [42], [43], [44]. These methods allow maximizing the value (1) with respect to ω, where ω denotes the parameters of a DNN that approximates the optimal policy function π_ω. A very popular DRL method that uses this idea is Trust Region Policy Optimization (TRPO) [45]. It solves the following optimization problem:

$$\begin{aligned}\max_{\omega} \quad & \mathbb{E}_{\pi_\omega,P}\left[\sum_{n=0}^{\infty}\gamma^n \frac{\pi_\omega(a|s)}{\pi_{old}(a|s)} A_{\pi_\omega}(s, a)\right], \\ s.t. \quad & \mathbb{E}_{\pi_{old},P}\left[D_{KL}\left(\pi_\omega || \pi_{old}\right)\right] \leq \delta\end{aligned} \tag{2}$$

where π_{old} refers to the value of the policy in the previous iteration., $D_{KL}(\pi_\omega||\pi_{old})$ is the Kullback-Leibler divergence between π_ω and π_{old}, δ is a threshold and $A_{\pi_\omega}(s, a)$ is the advantage function, used to estimate how good is action a when used in state s. Intuitively, in (2) we optimize the policy that maximizes the expected reward, where the maximum difference between the new policy with respect to the old one is limited by δ. This limitation is used to avoid large variations in the new policy, which may lead to poor performance: the new policy is bounded to lie within a region bounded by δ.

D. Partially Observable MDP

In many dynamical systems, the state s is unknown to the agent, who only has access to a partial or noisy observation o. This is known as Partially Observable MDP (POMDP) [38], and it is defined as an MDP adding:

- A set of observations O.
- An observation model $Z : S \times A \rightarrow O$, which denotes the probability of observing o' given that the system is in state s and the agent takes action a.

In POMDPs, the Markovian assumption does not hold: the optimal action depends on the whole history of previous observations and actions. There are several methods proposed to solve POMDPs, as shown in [38], [46] and [47]. It is also possible to use DRL tools to solve them in at least two possible ways [48]. The first consists in using recurrent DNNs, which allow incorporating the whole history of observations and actions. The second consists in using as input to the policy DNN a truncated history of fixed length, and then using the MDP tools already described to solve it. The latter approach can provide very good results [43], and it is followed in this work.

III. Inverse Reinforcement Learning

Inverse RL (IRL), also known as Inverse Optimal Control, is the complementary situation to RL: IRL tries to infer the reward that best explains the given policy of an agent, usually called expert. IRL can be used to model an unknown reward function [49] or to model an agent [50]. In IRL, we are given a set of trajectories $\zeta_k = \{(s_0, a_0), (s_1, a_1)...\}$, composed of state-action pairs, and we want to obtain the reward function r_θ that best explains the trajectories, where θ are the reward approximation parameters.

A. Maximum entropy inverse reinforcement learning

In the seminal paper [49], several linear methods are proposed to obtain r_θ function in simple IRL problems. However, these methods assume that the expert policy is optimal in all circumstances, which needs not be the case in real-life problems. To address this, the Maximum Entropy Principle (MEP) [51] was used for deterministic [52] and non-deterministic [53] policies. In both cases, the maximum causal entropy distribution is a Boltzmann distribution, where the reward is approximated as linear. However, these linear methods do not usually work well in practice. It is possible to apply the MEP ideas also to non-linear reward functions. In this case, again, we obtain a Boltzmann distribution [54] for the trajectories:

$$P_\theta(\zeta_k) = \frac{1}{Z}\exp(r_\theta(\zeta_k)), \tag{3}$$

where Z is the partition function. To obtain θ, let us assume that we have a non-linear policy function parameterized by ω. IRL means obtaining the reward r_θ that best explains the behavior of an expert, that is, the policy maximizes the total expected reward. Mathematically, this means [54]:

$$\max_\theta \left[\left(\max_\omega H(\pi_\omega) + \mathbb{E}_{\pi_\omega}[r_\theta(s,a)] \right) + \mathbb{E}_{\pi^*}[r_\theta(s,a)] \right], \tag{4}$$

where $\mathbb{E}_\pi[r_\theta(s,a)]$ denotes the expected reward under policy π, i.e., the value function (1), and $H(\pi)$ is the causal entropy of policy π. We can optimize iteratively:

1) We update ω, so that the policy π_ω both maximizes the causal entropy of ζ_k and r_θ, i.e., we use π_ω to define a policy similar to the expert.
2) We update θ so that the difference between the reward induced by the expert policy π^* and the reward induced by the policy π_ω is maximized. In other words, we try to find a reward function that separates as much as possible the behaviors of π^* and π_ω.

B. Generative Adversarial Imitation Learning

This iterative method, however, has to use an RL algorithm in the first step to obtain a policy, which is computationally expensive when having large S and A. An efficient way to solve (4) is Generative Adversarial Imitation Learning (GAIL) [55], which uses DNNs to approximate the policy and the reward functions and solves (4) using a Generative Adversarial Network (GAN) [56].

A GAN is a generative model which trains two DNNs in an adversarial fashion. The first DNN is called generator: it takes as input a random noise and produces an output which tries to match a certain distribution. The second DNN is called discriminator: it tries to discriminate between samples coming from the generator and the actual distribution. In GAIL, the generator approximates a policy π_ω: it takes as input a state s and outputs the probability of using any action $a \in A$. The discriminator is parameterized by θ and takes as input a tuple (s, a) and outputs the probability that the input was generated by the expert policy or π_ω. The reward function is:

$$r_\theta = -\log(D_\theta(s,a)). \tag{5}$$

By training the GAN, GAIL obtains a saddle point (π_ω, D_θ) of the following expression:

$$\mathbb{E}_{\pi_\omega}[\log(D_\theta(s,a))] + \mathbb{E}_{\pi^*}[\log(1 - D_\theta(s,a))] - \nu H(\pi_\omega), \tag{6}$$

where $\nu \geq 0$. Note that (6) is equivalent to (4) with minor implementation changes. First, GAIL uses a step of TRPO to minimize (6) with respect to ω: this step makes π_ω similar to the expert policy. GAIL relies crucially on TRPO, as the new policy is constrained to be close to the previous one (2), so that divergence due to noise in the gradient estimation is prevented. Second, GAIL uses an Adam [57] step with respect to θ to maximize (6).

GAIL presents several advantages. First, it uses DNNs to approximate both the reward and policy, which allows these functions to be very expressive, as DNNs are universal function approximators [40]. This means that GAIL could imitate arbitrarily complex behaviors from the expert. Second, GAIL can be used in large, even continuous, state and action spaces, as TRPO is able to deal with them. Third, GAIL is computationally efficient, since we need not solve an RL problem in each policy improvement step. Due to these properties, we use GAIL as IRL method in this work.

IV. Problem description

Let us now describe the attack setup. We have a wireless network that uses CSMA/CA in the MAC layer, with n_1 Good Stations (GSs) that respect the backoff procedure, and n_2 Attacking Stations (ASs) which do not respect the backoff procedure. For simplicity, we consider that the backoff procedure is the binary backoff mechanism described in IEEE 802.11 standard [58]. When the channel is idle, in order to avoid a collision due to several stations transmitting at once, each station starts a backoff counter which decrements while the channel is idle. When the counter of a station reaches 0, the station transmits. The counter is initialized following a uniform random variable in the interval $[0, CW]$, where CW is the backoff window. If a collision is detected, CW is duplicated, and if a transmission is successful, CW is divided by two. In all cases, $CW \in [CW_{min}, CW_{max}]$, where $CW_{max} = 2^\mu CW_{min}$ is the maximum size of the backoff window, μ is the maximum backoff stage and CW_{min} is the minimum size of the backoff window.

We assume a star topology in the network, where all stations communicate with a central node where the defense mechanism is located, which only observes the backoff window used by each station in past transmissions [24]. In order to estimate whether a station follows the binary exponential procedure or not, a Cramer-von Mises statistical test [59] can be used: the result of this test is the reputation t of each station. The test parameters are K, the number of backoff observations from the station under test, L, a number of samples obtained from the backoff distribution following the established procedure, and λ, a threshold on the reputation t that discriminates between ASs and GSs. For more details on this defense mechanism, we refer the reader to [26].

V. Intelligent attacker description

Let us now describe an intelligent attacker, based on DRL, which is able to exploit the backoff defense mechanism explained in Section IV as shown in [26]. The attacker solves a POMDP using TRPO, where the state of the defense mechanism is the reputation of each station, t, which is unknown to them. At each time step n, each of the n_2 ASs receives an observation o_i and it uses $\pi_\omega(o_i)$ to select its action a_i: it then receives a reward r and the next observation, o'_i. Thus:

- Each time step n corresponds to a backoff period, i.e., each n corresponds to a moment in which each station may transmit or not.
- A is formed by two discrete actions: transmit and not transmit.
- The reward is -1 if a GS starts transmitting at time step n, and 0 otherwise. Note that this reward models the target of the backoff attack: that GSs transmit less than ASs.
- Each observation contains the normalized time difference between the current time step and the last K transmissions and a flag indicating whether that transmission was successful or collided, as well as a flag indicating whether the station has been banned from the network.

We also consider the case in which there are several ASs that attack coordinately. We assume that ASs are able to communicate, and in this case, each o_i is built by concatenating the local observation of each agent, the mean of the observations sent by the other ASs and the mean of the observations of the GSs. Using the mean makes the concatenated vector invariant to the order and number of stations.

The intelligent attacker uses TRPO in order to obtain an optimal policy $\pi_\omega(o)$ such that it maximizes its cumulative reward. In case of having several agents, we use the training procedure described in [60] and [61]: a single policy is trained using data from all ASs, and then each AS executes a copy of the centralized policy.

VI. Intelligent defense mechanism description

We now propose an intelligent defense mechanism, which is able to cope with the intelligent attacker from Section V. To present a defense mechanism as general as possible, we use IRL tools, namely, GAIL. The main idea behind our method consists in using IRL on GSs to obtain the reward function $r_\theta(s,a)$ that explains the behavior of the GSs. Since ASs have a different reward function, the distributions of rewards will differ between ASs and a GSs; and hence, it will be possible to detect the ASs.

Note that our approach presents very significant advantages. First, by using GAIL, we do not need to analytically model the reward function of GSs explicitly as GAIL approximates it. Second, GAIL is a model-free method, thus we do not need to model the transition probability function, which may be very complex, as shown in [25]; we only need a simulator of the dynamical system. And third, GAIL is an IRL method efficient and accurate. We propose two possible ways to use GAIL in our problem: a method in which GAIL is trained prior to any network interaction that we call offline defense mechanism, and a second method in which GAIL is also trained during the interactions among the network stations, which we call online defense mechanism.

A. *Offline defense mechanism*

Before starting the interactions in the network, we train GAIL using ζ_{GS}, a set of trajectories in which all stations are GSs, to obtain the reward function $r_\theta(s,a)$ that maximizes $r_{GS} = r_\theta(\zeta_{GS})$, that is, the reward of GSs. Then, if we are given ζ_m, a set of trajectories from station m with unknown type (station m could be a GS or AS), we can obtain $r_m = r_\theta(\zeta_m)$. Note that both r_{GS} and r_m are two sets of rewards: if the values of r_m are similar to those of r_{GS}, then m is likely to be a GS, and an AS otherwise. In order to decide, we propose using a statistical test using the empirical CDF of r_{GS}. We define η as the α-percentile over the empirical CDF of r_{GS}. Mathematically, given $\alpha \in [0,1]$, we have that η is:

$$\begin{aligned} \eta = \arg\min_{r_{GS}} & CDF\left(r_{GS}\right) \\ s.t. \quad & CDF\left(r_{GS}\right) \geq \alpha \end{aligned}, \tag{7}$$

where we note that α controls the tradeoff between false alarm probability and power of the test.

Then, when we have interactions in the network, we collect ζ_m for each station m, where we consider that ζ_m is composed by j state-action pairs, and proceed to test. First, we obtain $r_m = r_\theta(\zeta_m)$, the rewards for station m assuming that station m is a GS. Then, we compute $i \leq j$: the number of times that $r_m \leq \eta$. A low i indicates that ζ_m comes from a GS, while a large i provides evidence that ζ_m comes from an AS.

Let us further assume that r_m are i.i.d. This assumption needs not be true, as consecutive state-action pairs are correlated through the transition probability function $\mathcal{P}$. However, we follow this assumption because it simplifies our model, allows using simpler calculations and does not require to know $\mathcal{P}$ for the correlation. Thus, i follows a Binomial distribution $i \sim B(j,\alpha)$, where j is the number of experiments (i.e., the number of elements of r_m) and α is the probability of $j=1$ (i.e., $r_m \leq \eta$, assuming that m is a GS). Observe that if m is a GS, i should be low, and if m is an AS, i should be high, since r_θ, obtained using GAIL, maximizes the rewards for the policy of GSs, and hence, r_m is low if m is an AS. Then, our decision rule is:

$$\begin{cases} \text{Station is GS} & \text{if} \quad i < \rho \\ \text{Station is AS} & \text{if} \quad i \geq \rho \end{cases}, \tag{8}$$

where ρ is a threshold that can be obtained as follows:

$$\begin{aligned} \rho = \arg\min_{k\in 0,\dots,j} \mathcal{L} = \arg\min_{k\in 0,\dots,j} \binom{j}{k} \alpha^k (1-\alpha)^{j-k} \\ s.t. \quad \mathcal{L} \geq 1-\alpha \end{aligned}, \tag{9}$$

where $\mathcal{L}$ is the Binomial distribution probability mass function. Note that k, in (9) is the number of successes, and hence, is equivalent to i: the number of times that $r_m \leq \eta$. Thus, we are modeling the probability of obtaining k values of reward below η. The constraint is the threshold of our test: we use the same α value, although a different value could be used, in order to set ρ as the threshold to decide that lower values of i are generated by GSs, but higher ones are not, with a confidence level of α.

A flow diagram for both the training and classification stages can be observed in Figures 2 and 3. The whole procedure can be observed in Figure 4, where we note that we train GAIL only once, but the classification stage may be run more than once.

Note that the decision method we propose, based on η and ρ, could be replaced by other decision methods, such as measuring the Kullback-Leibler divergence between the distribution r_{GS} and r_m, or other statistical tests [62]. In this work, we only focus on using η and ρ because it is a simple and computationally fast method which, nonetheless, provides good results.

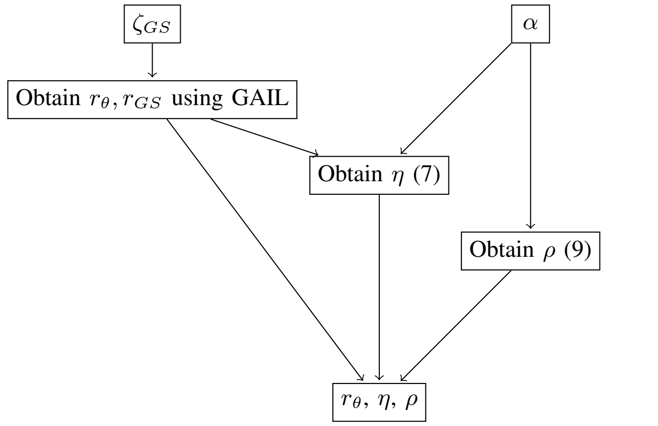

ζ_{GS}
α
Obtain r_θ, r_{GS} using GAIL
Obtain η (7)
Obtain ρ (9)
r_θ, η, ρ

Fig. 2. Flow diagram for the training stage of the proposed defense mechanism, both for online and offline cases.

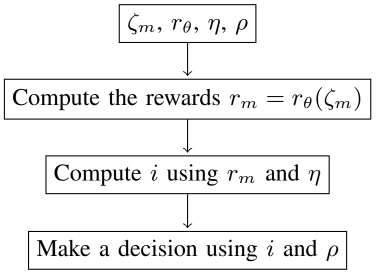

ζ_m, r_θ, η, ρ
Compute the rewards $r_m = r_\theta(\zeta_m)$
Compute i using r_m and η
Make a decision using i and ρ

Fig. 3. Flow diagram for the classification stage of our proposed defense mechanism, for both online and offline cases.

B. Online defense mechanism

The offline defense mechanism already described presents a potential problem: the behavior of GSs could be influenced by the actions of ASs. As GAIL is trained offline, using trajectories ζ_{GS} from a network in which there are only GSs, the same GSs in presence of ASs may present a different behavior as a consequence of the actions of the ASs: note that the behavior of GSs and ASs is coupled, as they affect each other. This could cause that some GSs are detected as ASs. In order to address this potential problem, we propose continuously training our defense mechanism by rerunning the training phase using trajectories from trusted GSs: this means that the GAIL classifier is updated continuously taking into account the effect of the ASs over the GSs. Note that the trusted GSs used for training GAIL must be known a priori, and their behavior is used to classify the rest of stations. This is what we denote as online defense mechanism, and a flow diagram describing it can be seen in Figure 5. Note that the main difference with respect to the offline case is that now, GAIL is updated during the interaction phase using state-action pairs from the trusted GSs. We note that the online defense mechanism has a higher computational complexity, as now GAIL is updated several times; but it also explicitly takes into account the effect of the ASs over the GSs, and hence, it can provide better results.

C. Discussion

Both of our proposed defense mechanisms rely on three assumptions: (1) the interaction between each station and the central node can be modeled using the MDP / POMDP framework, where sequences of states / observations and actions can be obtained and we have access to a simulator of the system, (2) GAIL can be used to obtain a solution to the IRL problem of the GSs, and (3) the behavior of ASs is different from the GSs in terms of IRL reward. These three assumptions are very general, and hence, they allow dealing in a very general way with unknown attacks. The third assumption is related to the fact that GAIL searches for a reward function that explains the trajectories: an AS behaves differently, and hence, we can use this reward function to discriminate between GSs and ASs.

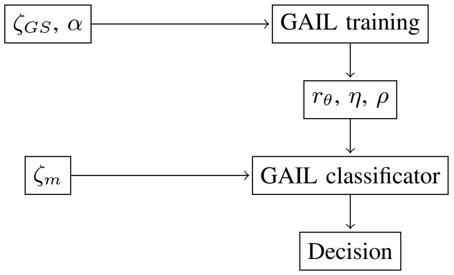


Fig. 4. Flow diagram for the offline defense mechanism, where the training stage is explained in Figure 2 and the classification stage is explained in Figure 3. Note that GAIL is trained once and offline, while there might be multiple decision: the thresholds obtained by GAIL are used each time that a decision is made.

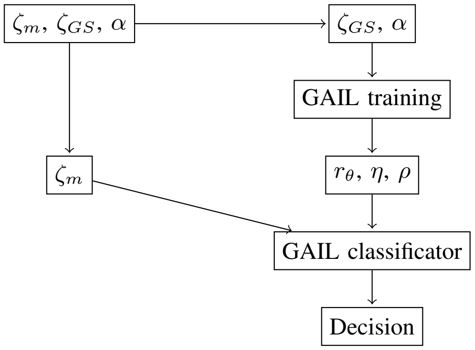


Fig. 5. Flow diagram for the online defense mechanism, where the training stage is explained in Figure 2 and the classification stage is explained in Figure 3. Note that the main difference with respect to the offline case in Figure 4 is that now GAIL is trained more than once, using ζ_{GS} collected from trusted GSs. Note that the thresholds obtained by GAIL are updated every time that GAIL is updated, whereas in the offline case the thresholds were fixed after the initial training.

Our approaches also present several weaknesses. The first one is that we do not take into account additional information about the ASs, as ours are general defense mechanisms. The more we know about the attack, the more an ad-hoc defense mechanism can be used. However, ad-hoc defense mechanisms could be exploited by minor attack variations [25]. We present general defense mechanisms, potentially valid against a broad set of attackers. Note that our defense mechanisms could also be used together with other defense mechanisms in order to incorporate more information about the attacker: we explore this idea in Section VII.

A second weakness is derived from the computational complexity associated with our methods. Even though GAIL is an efficient IRL method, nonetheless it has a significant resource consumption. Note that this is a problem that affects specially to the online defense mechanism, as it needs to train GAIL several times. Hence, there is a trade-off between a defense mechanism general enough and the computational load required. A final weakness affects only the online method: if the trusted GSs used to update GAIL are compromised, the defense mechanism may be exploited.

Our defense mechanisms also present several advantages. The first one is that they are very general and require very little knowledge about the concrete setup. The second is that we train only using GSs, so our methods are able to detect, potentially, any AS that presents a different behavior from GSs. And finally, observe that we need not model the transition probability function, as GAIL only needs access to a simulator of the system we want to defend.

VII. Empirical results: the partially observable backoff attack

We validate our defense mechanism from Section VI in the backoff attack setup described in Section IV, facing our defense mechanism to the intelligent attacker from Section V. The policy of the ASs is a feedforward DNN which takes as input the observation o_i and has three layers: the first two layers have 256 neurons and use rectified linear units activations, i.e., $f(x) = \max(0, x)$. The last layer of the policy DNN outputs the actions of the agent, i.e., whether to transmit in the current time step or not.

We test for 10 GSs and $\{1, 5\}$ ASs, in order to study the influence of communication among ASs in the attack. Thus, in total we have $\{11, 15\}$ stations in our network. As defense mechanism we use the scheme described in Section IV, with parameters $\lambda = 0.5$, $K = 5$ and $L = 1000$. The backoff mechanism implementation follows the values of the 802.11 IEEE standard [58]. Each episode simulates the backoff environment for $5 \cdot 10^5$ μs. We train the ASs using 200 TRPO iterations, where each TRPO policy update uses 2500 time steps.

We incorporate our proposed defense mechanisms as an additional security layer to the defense mechanism described in Section IV. Note that our security mechanism only observes the transmission times of a station, i.e., the time steps in which station m transmitted. This observation is the input to the reward DNN estimator that is trained using GAIL. We use a DNN with two hidden layers with 256 neurons each and hyperbolic tangent activation function. Note that the attacker is solving a POMDP by incorporating the observation information about the last $K = 5$ transmissions; our defense mechanism also uses a truncated history and keeps a record of the last 5 transmissions of each station.

For the training phase of GAIL in the offline defense mechanism, we use the trajectories from 100 episodes of $5 \cdot 10^5$ μs. In each of these episodes, we consider that there are $\{11, 15\}$ GSs. Then, we use GAIL to obtain an estimation of the reward function using 10 GAIL iterations: we update 3 times the generator and once the discriminator in each of these iterations. We use 10 iterations as there was no further improvement by increasing the training time, and set $\alpha = 0.05$ in order to obtain η and ρ using (7) and (9).

In the online defense mechanism, we use the same classifier DNN as in the offline case, but the training phase differs. We use the state-action pairs of 5 trusted GSs and train GAIL at the end of each TRPO iteration of the attackers. Hence, each time that the ASs update their policy, GAIL updates the reward estimator.

We run our defense mechanisms together with the statistical test described in Section IV for three cases: a baseline in which we only use the statistical test described in Section IV, and the cases in which the statistical test is combined with the online and offline defense mechanisms respectively. Our defense mechanism classifier is run every time that there are 5 new state-actions pairs per station, and if a station is detected as an AS by either defense method, it is banned from the network. Finally, for each value of the ASs, we simulate using 10 different seeds, and we use a discount factor $\gamma = 0.995$.

The simulation results averaged on the best 5 seeds for the defense mechanism can be observed in Table I, where we show the total reward, the proportion of stations banned and the proportion of bits transmitted by each station, for all the cases tested. The results indicate the following:

- In terms of ASs reward, both of our defense mechanisms significantly outperform the baseline. Remember that this reward, defined in Section V, was -1 each time that a GS transmitted. Thus, note that a smaller reward means that GSs transmit more often, which is worse for the attacker, but better for the defense mechanism. Regardless of the number of ASs, the online mechanism outperforms the offline one. As we will show later, this is due to the fact that GSs behave differently when ASs are present, and hence, the offline mechanism performs worse than the online one, although better than the baseline.
- In terms of proportion of stations banned, note that we can ban an AS (true positive) or a GS (false positive). In terms of ASs banned (true positive rate), note that both online and offline improve the baseline, which was the main objective of our defense mechanisms. They are able to detect more ASs based only on analyzing the behavior of GSs. Note that the online method also offers higher accuracy than the offline one: as mentioned, this is due to the fact that online method is continuously analyzing the behavior of GSs. Regarding the proportion of GSs banned (false positive rate), the best method is the baseline, and then the online method. Again, the online method takes advantage of continuously analyzing the GSs behavior, to obtain a false positive rate half of the offline method.
- In terms of transmitted bits, let us recall that the backoff attack targets to reduce the bits transmitted by GSs, which means that ASs transmit more than GSs. In the baseline case, note how ASs have a proportion of bits transmitted up to nearly three times higher than GSs, illustrating the consequences of the backoff attack. However, both of our defense mechanisms reduce this gap, specially the online one, which is able to enforce a more fair use of the network resources.

		1 AS	5 ASs
Reward	Baseline	-40.64 ± 8.44	-19.59 ± 3.73
	Offline	-41.34 ± 6.60	$\mathbf{-21.83 \pm 4.44}$
	Online	$\mathbf{-81.19 \pm 2.86}$	$\mathbf{-28.57 \pm 4.44}$
Proportion ASs banned	Baseline	0.19 ± 0.39	0.60 ± 0.26
	Offline	$\mathbf{0.64 \pm 0.48}$	$\mathbf{0.73 \pm 0.27}$
	Online	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.86 \pm 0.19}$
Proportion GSs banned	Baseline	$\mathbf{0.002 \pm 0.01}$	$\mathbf{0.001 \pm 0.01}$
	Offline	0.27 ± 0.23	0.17 ± 0.14
	Online	0.17 ± 0.12	0.08 ± 0.12
Proportion bits per AS	Baseline	0.16 ± 0.05	0.11 ± 0.02
	Offline	0.12 ± 0.05	$\mathbf{0.10 \pm 0.02}$
	Online	$\mathbf{0.04 \pm 0.01}$	$\mathbf{0.07 \pm 0.02}$
Proportion bits per GS	Baseline	0.08 ± 0.01	0.04 ± 0.01
	Offline	$\mathbf{0.09 \pm 0.01}$	$\mathbf{0.05 \pm 0.01}$
	Online	$\mathbf{0.10 \pm 0.01}$	$\mathbf{0.06 \pm 0.01}$

TABLE I

Final results obtained averaging 100 episodes for each of the best 5 seeds after training. We show the mean final value, ± one standard deviation. Bold entries are the values with best mean, where a Welch test is used to detect whether means are significantly different for a significance level of 0.01 with respect to the baseline. The metrics used are the TRPO reward for ASs, the proportion of stations banned from the network, and the proportion of bits transmitted per station.

Thus, our proposed defense mechanisms outperform the baseline, specially the online method. The only difference between online and offline methods are that the online method keeps continuously updating η, and hence, it adapts to the change in GSs behavior due to ASs, since the actions of ASs (transmit or not) do affect to GSs. Figure 6 shows the rewards distribution for the offline method. Observe how the distribution of r_{GS} changes after the introduction of a single AS: even though the change seems small, note how the critic part of the histogram is the left queue, used to define the η threshold value. Small changes in that region mean that GSs will be wrongly classified as ASs, and hence, banned. The online method, on the contrary, updates η continuously, and hence, it has a lower

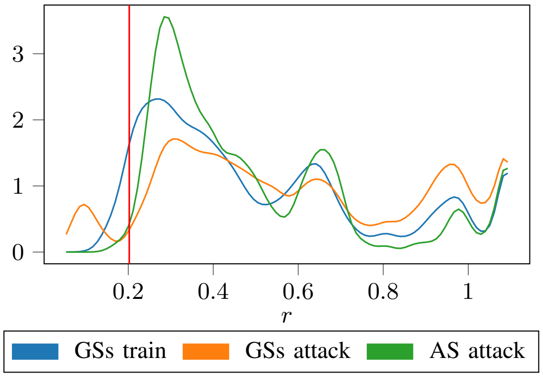


Fig. 6. Distribution of rewards in the offline method for GSs without attack (i.e., during training), under attack (i.e., with 1 AS) and the rewards of the AS. The vertical red line is η, the reward threshold value. Observe that the AS puts more weight in lower values of the reward, and also, note how the introduction of a single AS changes the distribution below η (the red line).

probability of misclassifying stations (see Table I). Finally, note also how the AS in Figure 6 does pose a challenge to be detected, as it mimics quite good the behavior of GSs. Thus, intelligent attackers pose a significant threat to current defense mechanisms that may not be prepared to deal with them.

VIII. Conclusions

In this work, we have given a step forward towards intelligent defense mechanisms that are able to cope with intelligent attackers based on DRL tools. As we have seen in our simulations, intelligent attackers are able to exploit unknown defense mechanisms simply by interacting with them, and we can think of such attackers as automatic exploit discoverers, whose success has already been shown [25], [26].

In order to address this challenge, we have proposed two different defense mechanisms based on IRL, that make little assumptions about the attack used, and hence, can potentially adapt to a wide set of attacks and attackers. Our results show that both mechanisms are successful in detecting intelligent attackers. Although our methods require a significant computational effort, we propose an offline mechanism, that can be trained offline and then deployed. However, this offline mechanism does not adapt to the effect that ASs have over GSs, and we propose another online method that obtains remarkable results, at a higher computational cost.

This work poses several interesting future lines. First, the classifier proposed is simple and powerful, as it detects many ASs, but at the same time, it has a non negligible false positive rate that affects GSs: thus, finding a classifier that improves the false positive rate is one line of work. Note that we could use tools based on anomaly detection tools [63], other statistical tests [62], or distribution-related metrics, such as the Kullback-Leibler divergence. And second, we have remarked that ASs affect GSs, as both types of stations are coupled. This situation could be modelled using dynamic game theory tools, which face the problem of computational complexity, although there is a lot of promising work ongoing in that direction [64].

Acknowledgment

This research was funded by the Spanish Ministry of Science and Innovation under the grant PID2020-112502RB-C41 (NAUTILUS). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

References

[1] K. Yang, *Wireless sensor networks*. Springer, 2014.

[2] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, “Wireless sensor networks: a survey on recent developments and potential synergies,” *The Journal of supercomputing*, vol. 68, no. 1, pp. 1–48, 2014.

[3] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, “Software defined networking for improved wireless sensor network management: A survey,” *Sensors*, vol. 17, no. 5, p. 1031, 2017.

[4] I. Tomić and J. A. McCann, “A survey of potential security issues in existing wireless sensor network protocols,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1910–1923, 2017.

[5] T. Gu, A. Abhishek, H. Fu, H. Zhang, D. Basu, and P. Mohapatra, “Towards learning-automation iot attack detection through reinforcement learning,” in *2020 IEEE 21st International Symposium on” A World of Wireless, Mobile and Multimedia Networks”(WoWMoM)*. IEEE, 2020, pp. 88–97.

[6] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[7] Y. Shi, Y. E. Sagduyu, T. Erpek, K. Davaslioglu, Z. Lu, and J. H. Li, “Adversarial deep learning for cognitive radio security: jamming attack and defense strategies,” in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[8] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, “Iot security techniques based on machine learning,” *arXiv preprint arXiv:1801.06275*, 2018.

[9] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, “Machine learning in wireless sensor networks: Algorithms, strategies, and applications,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014.

[10] J. Cannady, “Next generation intrusion detection: Autonomous reinforcement learning of network attacks,” in *Proceedings of the 23rd national information systems security conference*, 2000, pp. 1–12.

[11] Y. Gwon, S. Dastangoo, C. Fossa, and H. Kung, “Competing mobile network game: Embracing antijamming and jamming strategies with reinforcement learning,” in *Communications and Network Security (CNS), 2013 IEEE Conference on*. IEEE, 2013, pp. 28–36.

[12] L. Xiao, Y. Li, G. Liu, Q. Li, and W. Zhuang, “Spoofing detection with reinforcement learning in wireless networks,” in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–5.

[13] Y. Li, D. E. Quevedo, S. Dey, and L. Shi, “Sinr-based dos attack on remote state estimation: A game-theoretic approach,” *IEEE Transactions on Control of Network Systems*, vol. 4, no. 3, pp. 632–642, 2017.

[14] L. Xiao, C. Xie, T. Chen, H. Dai, and H. V. Poor, “A mobile offloading game against smart attacks,” *IEEE Access*, vol. 4, pp. 2281–2291, 2016.

[15] L. Xiao, Y. Li, X. Huang, and X. Du, “Cloud-based malware detection game for mobile devices with offloading,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2742–2750, 2017.

[16] M. A. Aref, S. K. Jayaweera, and S. Machuzak, “Multi-agent reinforcement learning based cognitive anti-jamming,” in *Wireless Communications and Networking Conference (WCNC), 2017 IEEE*. IEEE, 2017, pp. 1–6.

[17] G. Han, L. Xiao, and H. V. Poor, “Two-dimensional anti-jamming communication based on deep reinforcement learning,” in *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing,*, 2017.

[18] A. G. Fragkiadakis, E. Z. Tragos, and I. G. Askoxylakis, “A survey on security threats and detection techniques in cognitive radio networks,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 428–445, 2013.

[19] P. Sengar and N. Bhardwaj, “A survey on security and various attacks in wireless sensor network,” *International Journal of Computer Sciences and Engineering*, vol. 5, no. 4, pp. 78–84, 2017.

[20] L. Zhang, G. Ding, Q. Wu, Y. Zou, Z. Han, and J. Wang, “Byzantine attack and defense in cognitive radio networks: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1342–1363, 2015.

[21] J. Wu, Y. Yu, H. Zhu, T. Song, and J. Hu, “Cost-benefit tradeoff of byzantine attack in cooperative spectrum sensing,” *IEEE Systems Journal*, vol. 14, no. 2, pp. 2532–2543, 2019.

[22] K. Lakshmi Narayanan, R. Santhana Krishnan, E. Golden Julie, Y. Harold Robinson, and V. Shanmuganathan, “Machine learning based detection and a novel ec-brtt algorithm based prevention of dos attacks in wireless sensor networks,” *Wireless Personal Communications*, pp. 1–25, 2021.

[23] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, “A survey on jamming attacks and countermeasures in wsns,” *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, 2009.

[24] W. Wang, Y. Sun, H. Li, and Z. Han, “Cross-layer attack and defense in cognitive radio networks,” in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–6.

[25] J. Parras and S. Zazo, "Learning attack mechanisms in wireless sensor networks using markov decision processes," *Expert Systems with Applications*, vol. 122, pp. 376–387, 2019.

[26] J. Parras, M. Hüttenrauch, S. Zazo, and G. Neumann, "Deep reinforcement learning for attacking wireless sensor networks," *Sensors*, vol. 21, no. 12, p. 4060, 2021.

[27] M. Zolotukhin, S. Kumar, and T. Hämäläinen, "Reinforcement learning for attack mitigation in sdn-enabled networks," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2020, pp. 282–286.

[28] Y. Chen, S. Huang, F. Liu, Z. Wang, and X. Sun, "Evaluation of reinforcement learning-based false data injection attack to automatic voltage control," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2158–2169, 2018.

[29] Z. Ni and S. Paul, "A multistage game in smart grid security: A reinforcement learning solution," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2684–2695, 2019.

[30] Z. Wang, H. He, Z. Wan, and Y. Sun, "Coordinated topology attacks in smart grid using deep reinforcement learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1407–1415, 2020.

[31] M. N. Kurt, O. Ogundijo, C. Li, and X. Wang, "Online cyber-attack detection in smart grid: A reinforcement learning approach," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5174–5185, 2018.

[32] F. Wei, Z. Wan, and H. He, "Cyber-attack recovery strategy for smart grid based on deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2476–2486, 2019.

[33] C. Roberts, S.-T. Ngo, A. Milesi, S. Peisert, D. Arnold, S. Saha, A. Scaglione, N. Johnson, A. Kocheturov, and D. Fradkin, "Deep reinforcement learning for der cyber-attack mitigation," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2020, pp. 1–7.

[34] M. Li, Y. Sun, H. Lu, S. Maharjan, and Z. Tian, "Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6266–6278, 2019.

[35] J. Parras and S. Zazo, "Wireless networks under a backoff attack: A game theoretical perspective," *Sensors*, vol. 18, no. 2, p. 404, 2018.

[36] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific, 2005, vol. 1.

[37] ——, *Dynamic programming and optimal control*. Athena Scientific, 2007, vol. 2.

[38] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[39] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[40] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[41] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[44] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32. Bejing, China: PMLR, 22–24 Jun 2014, pp. 387–395.

[45] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[46] M. L. Littman and R. S. Sutton, "Predictive representations of state," *Advances in neural information processing systems*, pp. 1555–1561, 2002.

[47] S. P. Singh, M. L. Littman, N. K. Jong, D. Pardoe, and P. Stone, "Learning predictive state representations," *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 712–719, 2003.

[48] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *Proc. of Conf. on Artificial Intelligence, AAAI, 2015.*, 2015.

[49] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *ICML*, vol. 1, 2000, p. 2.

[50] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[51] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, pp. 620–630, 1957.

[52] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[53] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Omnipress, 2010, pp. 1255–1262.

[54] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *arXiv preprint arXiv:1611.03852*, 2016.

[55] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[58] IEEE, "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," pp. 1–3534, December 2016.

[59] T. W. Anderson, "On the distribution of the two-sample cramer-von mises criterion," *The Annals of Mathematical Statistics*, pp. 1148–1159, 1962.

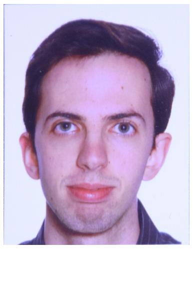
[60] A. Šošić, W. R. KhudaBukhsh, A. M. Zoubir, and H. Koeppl, "Inverse reinforcement learning in swarm systems," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 1413–1421.

[61] M. Hüttenrauch, A. Šošić, and G. Neumann, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-476.html

[62] M. Hernandez, G. Epelde, A. Alberdi, R. Cilla, and D. Rankin, "Standardised metrics and methods for synthetic tabular data evaluation," 2021.

[63] Y. Wang, J. Wong, and A. Miner, "Anomaly intrusion detection using one class svm," in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.* IEEE, 2004, pp. 358–364.

[64] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," *arXiv preprint arXiv:1707.09183*, 2017.



Juan Parras received his B.S in Telecommunications Engineering from Universidad de Jaén in 2014, and MSC and PhD in Telecommunications Engineering from Universidad Politécnica de Madrid (UPM) in 2016 and 2020. He is currently an Assistant Professor at UPM and his research interests include audio and radio signal processing, game theory and deep reinforcement learning applied to communications networks.

Alejando Almodóvar received his bachelor's degree and MSc in Telecommunications Engineering from Universidad Politécnica de Madrid (UPM) in 2020 and 2022. He is working as a researcher at UPM since 2021, with the goal of obtaining a PhD focused on signal processing and machine learning applications.

Patricia A. Apellániz completed her Bachelor's degree program in Telecommunication Engineering by the Universidad Autónoma de Madrid (UAM) in 2018 and, later, she received her Master's degree in Telecommunication Engineering at the Universidad Politécnica de Madrid (UPM). She is working towards her PhD at UPM on deep learning algorithms for medical applications and is also a researcher in this field. Her research interests go from audio and image signal processing to different deep learning applications.

Santiago Zazo is Dr. Engineer by the Universidad Politécnica de Madrid (UPM) in 1995. In 1998 he joined UPM, where currently he is Professor in Signal Theory and Communications. His main research activities are in the field of Signal Processing. More recently, he has been mostly focused on distributed optimization, optimum control, game theory and reinforcement learning. He is author / coauthor of more than 40 journal papers and about 200 conference papers.